Adaptive Block Pinning for Multi-core Architectures

Rakesh Kumar[1], Nitin Chaturvedi[1], T S B Sudarshan[2]
[1]Student, Electrical and Electronics Engineering Group
[2]*Assistant Professor, Computer Science & Information Systems Group*
Birla Institute of Technology and Science, Pilani
{h2006119, nitin80, tsbs}@bits-pilani.ac.in

Abstract

Difference between speed of processor and memory is increasing with advent of every new technology. Chip Multi Processors (CMP) have further increased the load on the memory hierarchy. So it has become important to manage on-chip memory judiciously to reduce average memory access time. The previous research has shown that it is better to have a shared cache at the last level of on-chip memory hierarchy. Sharing last level of cache gives rise to a new category of cache misses; those were not present in uniprocessor, called "inter-processor misses". This paper proposes a technique to eliminate inter-processor misses by giving replacement ownership of a block to a processor who brought it into the cache. This reduction in inter-processor misses, which constitutes 40% of over all misses, will result in performance improvement. Also two different ways of relinquishing the ownership of a block are being proposed, so that if some other processor, other than owner, can make use of the block in a more efficient way, ownership will be transferred to the new processor.

1. Introduction

In CMP, last level of on-chip memory can be organized as either shared or private cache. Private caches have the advantage of low access latency but these caches fail to make optimum use of on-chip memory space because some blocks may need to be replicated. While shared caches make optimum use of on-chip cache space, they suffer from high access latency compared to private caches.

Authors: Rakesh Kumar, Nitin Chaturvedi

L Hsu [3] has shown that organizing last level cache as shared cache gives better performance than private caches. Organizing last level cache as shared cache gives rise to another type of misses that were not present in the private caches: "inter-processor misses". A miss is called inter-processor miss, in a dual core system with cores P1 and P2, when P2 evicts a block which was brought into the cache by P1 and due to this eviction P1 suffers a miss and vice versa. As shown in fig.1 inter-processor misses constitutes about 40% of over all misses. So, it is a worthwhile goal to reduce these misses. To eliminate inter-processor misses, Shekhar [1] gives replacement ownership of a set to a processor, who brings in the first block into that set and only this processor is allowed to evict the blocks from the set. Ownership is only for replacement; other processors can read and write into the set but can't evict the blocks.

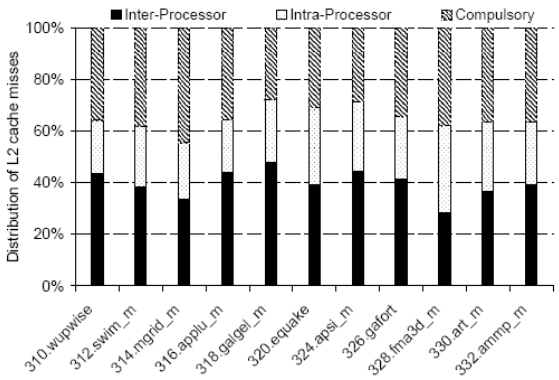

Fig 1) Distribution of Compulsory, Intra-processor and Inter-processor misses in L2 cache SPEComp Benchmarks [1].

This paper provides a fine control over the replacement ownership. Instead of giving ownership of complete set, this paper proposes a technique to provide ownership of individual blocks in a set and it will be shown that this fine

control will result in a better utilization of the blocks inside a set. "Processor Owned Private" (POP) caches were proposed by [1]. One POP cache is associated with each processor. Ownership is only in terms of replacements; any processor can read and write in any POP cache.

Rest of the paper is organized as follows: Related work is described in Section 2. Section 3 explains proposed architecture and ownership relinquishing techniques. Section 4 provides proposed implementation details and Section 5 concludes.

2. Related Work

Many researchers are extensively working on managing shared caches in Chip Multi Processors (CMP). M. Dubois [4] first introduced a class of misses that was not present in the uniprocessors. This category is called coherency misses, and is present only in Multi Processors. These misses occur because of invalidation of cache blocks shared between private caches of multiple processors. These misses can further be divided into true and false sharing misses.

Shekhar [1] introduced another way of categorizing misses in multiprocessors. This is known as CII misses. CII are compulsory misses, intra-processor misses and inter-processor misses. In proposed architecture, inter-processor misses are eliminated by giving replacement ownership of a block to a processor, while Shekhar eliminates inter-processor misses by giving replacement ownership of a set to a processor. For "hot set" [1] ownership of complete set is given to a single processor. But if set is not a hot set, giving ownership to single processor will increase load on the POP caches of other processors. As shown in fig 2, only about 9% of memory addresses results in hot sets, so number of hot sets is not going to be too large. As a result, most of the sets will not be owned by single processors, this releases load on POP caches.

Qureshi [2] divides number of blocks in a set among different processors. Here, at the end of a time frame, miss rate is measured, which means any action to reduce the growing miss rate can be taken only at the end of time frame. This paper proposes an implementation where corrective action can be taken at any time when miss rate grows above a given threshold value.

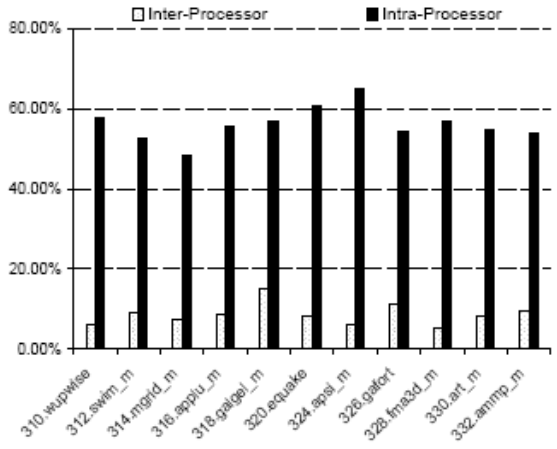

Fig 2) Memory addresses leading to Inter and Intra-processor misses [1].

3. Architecture

3.1 Block Ownership

Inter-processor misses occurs because a block brought into the cache by one processor is evicted by other processor and the original processor suffers a miss due to this eviction. A simple way to avoid these misses is to give ownership of a block to a processor, the one who brought the block into the cache, so that only the processor with ownership has the permission to evict the block. This can be done by defining replacement ownership of the block. For example, in a dual core processor with cores P1 and P2 where both P1 and P2 are generating addresses whose index bits correspond to this set. In the absence of block ownership, any processor, say P2, suffering a miss may evict a block from this set which belongs to P1, this eviction may cause a miss for P1 when next time it accesses the block which is just evicted by P2. If block replacement ownership is given to processors, P2 will not be able to evict a block owned by P1 and vice-versa. But eliminating inter-processor misses in this way may lead to an increase in intra-processor misses. Number of intra-processor misses will depend on whether the set is a "hot set" or not.

Giving block ownership in a hot set will increase intra-processor misses, since now a

processor has less number of blocks to choose from, to replace a block while it requires more number of blocks in that set. So, hot set miss rate is high either due to inter-processor misses or due to increased intra-processor misses. To reduce this increase in intra-processor misses, POP caches are used. If during last N access to a set there are M or more misses, where M is the threshold value, then the set is considered to be a "hot set" and the ownership of one of the processor is cancelled and the processor will now bring its blocks from memory to its POP cache instead of the hot set. This will reduce the traffic to hot set and eventually miss rate will come down. This process of canceling the ownership of processors from a particular set may lead to a situation where only one processor owns all the blocks in a set. Here the ownership of a processor is cancelled if it owns least number of blocks.

If a set is not a hot set, means not many addresses are being generated by processors with the index address of this set, then proper distribution of block ownership among the processors is necessary, to reduce the miss rate. Consider a case when P1 owns most of the blocks and rarely using some of these blocks while P2 has ownership of few blocks and suffering misses in that set because it has fewer blocks to choose from when evicting. If the ownership of less frequently used blocks of P1 is transferred to P2, over all miss rate can be reduced. Algorithm used for relinquishing the ownership is explained in section 3.4. Also by allowing all the processors to share "non-hot set" the load on the POP cache can be reduced.

3.2 Cache initialization

To give block ownership, (log *n*) bits in each block are needed to indicate owner of the block, where n is number of processors. When first time a processor brings a block from memory to cache, its id number will be written in the ownership bits of the block. Now only this as processor can evict the block from cache, as long it keeps ownership, not any other processor.

3.3 Cache HIT and MISS

Cache is organized as POP caches and a common cache. When any processor faces an L1 cache miss, in addition to common L2 cache, POP caches of all the processors are also checked for requested block. If there is a miss in common L2 cache and hit in one of the POP cache, request is served from POP cache. These two are non-inclusive in nature.

When a cache miss occurs, it may result in following scenarios:

1) Requested block address may point to a set where some of the blocks are not owned by any processor. In this case, requested block will be transferred from memory to the indexed set and ownership bits will be set with the id of the requesting processor.
2) Requested block address may point to a set where all the blocks are owned by processors other than the one with a miss. In this case, a block can not be replaced from this set because requesting processor doesn't own any block. So, data from memory will be transferred to the POP cache of the processor suffering miss.
3) Requested block address is pointing to a set where requesting processor owns some of the blocks. In this case processor will replace one of the blocks owned by it with the new block. Block to be replaced can be selected by LRU. In this case, block to be replaced is one which is least recently used blocks among the blocks owned by the processor, which need not be the least recently used block of the set.

3.4 Ownership Relinquishment

This paper proposes two methods to relinquish the ownership of a block:

In the first method, one counter per block is used. The counter is initialized to half of the maximum count. Every time when the block is accessed and results in a hit, counter value is increased by one. If counter reaches maximum value i.e. all 1's it will stay there. If a processor experiences a miss in a particular set, then counters corresponding to all the blocks owned by other processors are decremented by one. If any counter hits zero, ownership of this block is cancelled and given to the processor whose miss makes the counter to hit zero. Qualitatively, a counter hitting zero means that the processor owning it is not using it effectively and this block

can be used more effectively by other processor. This technique has a major drawback that numbers of counters required are equal to the number of blocks in the cache. This huge hardware requirement makes this technique less attractive.

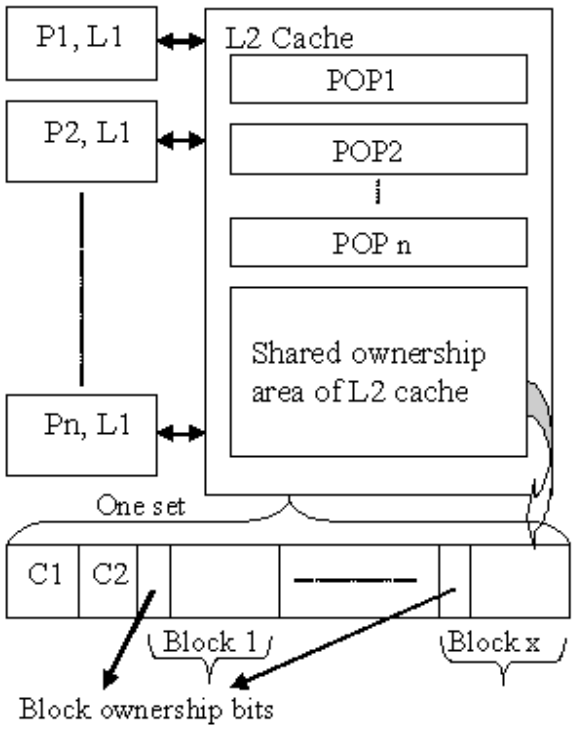

Fig 3) Block Diagram for proposed Architecture

The other technique for ownership relinquishment requires just two counters C1 and C2 per set as shown in fig 3. C2 is used to determine whether or not a set is a "hot set" and C1 is used to fine tune the number of block owned by each processor in a set. Selecting two counters is based on the observation that miss rate in a set can rise because of two reasons:

I) Set is a "hot set" and most of the processors are trying to put their blocks in the same set and hence intra-processor misses are more.

II) Set is not a hot set but distribution of blocks in the set is not proper, i.e. processor requiring more blocks owns less blocks and processor owning more blocks is not utilizing them.

Counter C1 produces a high output if there are X misses in last Y accesses to a set and counter C2 produces a high output if there are M misses in last N accesses to that set. Here M is a multiple of X and N is a multiple of Y. Multiplication factor in both cases is same. So, if miss rate increases above a particular value, C1 will detect it first, and the set is assumed not to be hot set at this point. The ownership of the blocks in the set which are not being utilized properly is cancelled. To do this, whenever C1 produces a high output, ownership of the least recently used block in the set is cancelled, so that a processor suffering more misses can acquire the ownership of this block and miss rate comes down. Qualitatively, in canceling ownership of least recently used block, it is assumed that this block is not being utilized properly by owner and is required by other processor than the current owner. Once ownership of a block is cancelled, C1 is reset to it's initial value, if miss rate still remains high after few such attempts, number of such attempts is determined by ratio of N to Y, C2 will also produce a high output and the set is treated as a hot set, means every processor is trying to put its blocks in this set. As stated earlier, the ownership of all the blocks of a particular processor is cancelled and this processor will now bring any new blocks to its POP cache instead of the "hot set". This cancellation of ownership of blocks will continue until either miss goes below the threshold value or complete set is owned by single processor. This will reduce the load on the hot set and miss rate will reduce.

4. Proposed Implementation

Proposed architecture will be simulated using Simics full system simulator [5]. In addition to Simics, General Execution-driven Multiprocessor Simulator (GEMS) [6] which is based on Simics, will be used for simulating the complete architecture.

The optimum values of the parameters used in the design will be evaluated using simulations. These parameters are size of C1 and C2 counters or in turn the values of N and Y, the number of access in which miss rate is to be

calculated. Another parameter is threshold value of miss rate and the value of M and X, the number of misses in N and Y accesses respectively. To find out optimum value of miss rate, applications can be run beforehand to get the information about miss rate as a function of number of blocks per set, and then decide, when does miss rate becomes insensitive to increase in number of blocks. Also, effective miss rate for different combinations of applications can be found out. Miss rate for different combinations of applications may vary, while initializing the system, operating system can change the value of miss rate threshold, depending upon which application mix is to be run. SPEC 2006 benchmark will be used in the simulations to find out values of these parameters.

5. Conclusion

Inter-processor misses constitutes 40% of total number of misses in a Chip Multi Processor with shared level 2 cache. This paper proposed a new architecture to eliminate these misses without a significant increase in intra-processor misses. Proposed architecture gives replacement ownership of a block to one of the processors and only owner can evict a block form cache, thus eliminate inter-processor misses.

This paper also showed that if a processor is not utilizing blocks owned by it optimally, ownership of block can be transferred to other processors. This paper showed two techniques to relinquish the ownership of a block. In future, better ways of selecting a processor to give ownership to, when ownership of a block is relinquished has to be investigated.

6. References

[1] Shekhar Srikantaiah, Mahmut Kandemir, Mary Jane Irwin. "Adaptive Set Pinning: Managing Shared Caches in Chip Multiprocessors." Proceedings of ASPLOS'08, ACM/IEEE, pages 135-144 March, 2008.

[2] M. K. Qureshi and Y. N. Patt. "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches." In *Proc. of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 423-432 , 2006.

[3] L. Hsu, R. Iyer, S. Makineni, S. Reinhardt, and D. Newell. "Exploring the cache design space for large scale cmps." *SIGARCH Computer Architecture News*, 33(4):pages 24–33, 2005.

[4] M. Dubois, J. Skeppstedt, L. Ricciulli, K. Ramamurthy, and P. Stenstrom. "The detection and elimination of useless misses in multiprocessors." In *Proc. of the 20th Annual International Symposium on Computer Architecture*, pages 88–97, San Diego, 1993.

[5] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. "Simics: A full system simulation platform." *Computer*, 35(2): pages 50–58, 2002.

[6] Milo M.K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood, "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset" *Computer Architecture News (CAN)*, pages 92-99, November 2005.